Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME			
CENTRE NUMBER		CANDIDATE NUMBER	

COMPUTER SCIENCE **9608/21**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2020**

2 hours

You must answer on the question paper.

No additional materials are needed.

INSTRUCTIONS
- Answer **all** questions.
- Use a black or dark blue pen.
- Write your name, centre number and candidate number in the boxes at the top of the page.
- Write your answer to each question in the space provided.
- Do **not** use an erasable pen or correction fluid.
- Do **not** write on any bar codes.
- You may use an HB pencil for any diagrams, graphs or rough working.
- Calculators must **not** be used in this paper.

INFORMATION
- The total mark for this paper is 75.
- The number of marks for each question or part question is shown in brackets [ ].
- No marks will be awarded for using brand names of software packages or hardware.

This document has **20** pages. Blank pages are indicated.

1 **(a)** Translation is one stage of the program development cycle.

State **three** other stages.

1 ....

2 ....

3 ....
[3]

(b) Define the following types of maintenance.

Corrective maintenance ....

....

....

Adaptive maintenance ....

....

....
[2]

(c) Experienced programmers have a **transferable skill**.

Explain how this skill might be useful for a programmer.

....

....

....

.... [2]

(d) Jackie has written a program and has used the identifier names `I1`, `I2`, and `I3`.

Explain why this is not good practice.

....

....

....

.... [2]

(e) A pseudocode algorithm assigns values to three variables as follows:

```
GateOpen ← FALSE
Alarm ← TRUE
PowerFail ← TRUE
```

Evaluate the expressions given in the following table:

Expression	**Evaluates to**
`Alarm OR NOT PowerFail`	
`NOT (Alarm AND PowerFail)`	
`(GateOpen OR Alarm) AND PowerFail`	
`(GateOpen AND Alarm) OR NOT PowerFail`	

[2]

2 **(a)** User names are stored in a text file. Each line of the file represents one name. Before a new user name can be issued, a check has to be made to ensure that the new name is unique.

Use **structured English** to describe an algorithm that would prompt and input a new user name and output a message to indicate whether or not it is unique.

....

....

....

....

....

....

....

....

....

....

....

....

....

....

.... [6]

(b) Complete the pseudocode expressions in the following table.

Use **only** the functions and operators described in the **Appendix** on pages 18–19.

Expression	**Evaluates to**
`........................ ("Stepwise" , ........................) & "art"`	`"Start"`
`........................ ("Concatenate" , ........................ , ........................)`	`"ten"`
`2 * ......................... ("Kipper")`	`12`
`TRUE ......................... FALSE`	`TRUE`
`......................... (9, 2)`	`1`

[5]

(c) Study the following pseudocode.

Line numbers are given for reference only.

```
01  PROCEDURE StringClean(InString : STRING)
02
03     DECLARE NextChar : CHAR
04     DECLARE OutString : STRING
05     DECLARE Index : INTEGER
06
07     OutString ← ""
08
09     FOR Index ← 1 TO LENGTH(InString)
10
11        NextChar ← MID(InString, Index, 1)
12        NextChar ← LCASE(NextChar)
13
14        IF NextChar >= 'a' AND NextChar <= 'z'
15           THEN
16              OutString ← OutString & NextChar
17        ENDIF
18
19     ENDFOR
20
21     OUTPUT OutString
22
23  ENDPROCEDURE
```

Complete the following table by entering an appropriate answer.

	Answer
The name for the type of loop used	
A line number of a selection statement	
The scope of `OutString`	
The name of a function that is called	
A line number containing a logical operator	

[5]

3 The procedure `OutputLines()` outputs a number of lines from a text file.

An example of the use of the procedure is given by the following pseudocode:

```
CALL OutputLines(FileName, StartLine, NumberLines)
```

Parameter	**Data type**	**Description**
`FileName`	`STRING`	The name of the text file
`StartLine`	`INTEGER`	The number of the first line to be output
`NumberLines`	`INTEGER`	The number of lines to be output

The procedure is tested using the file `MyFile.txt` that contains 100 lines of text.

The procedure gives the expected result when called as follows:

```
CALL OutputLines("MyFile.txt", 1, 10)
```

(a) The procedure is correctly called with three parameters of the appropriate data types, but the procedure does not give the expected result.

Give **three different** reasons why this might happen.

1 ....

....

2 ....

....

3 ....

....

[3]

(b) Write **program code** for the procedure `OutputLines()`.

Note: Parameter validation is **not** necessary.

Visual Basic and Pascal: You should include the declaration statements for variables.
Python: You should show a comment statement for each variable used with its data type.

Programming language ....

Program code

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

.... [7]

(c) A program is compiled without producing any errors.

(i) Describe **one** type of error that the program could still contain.

....

....

....

.... [2]

(ii) Give **two** techniques that may be used to identify an error of the type given in **part (c)(i)**.

Technique 1 ....

....

Technique 2 ....

....
[2]

(d) State **two** reasons why the use of library subroutines can be a benefit in program development.

1 ....

....

2 ....

....
[2]

BLANK PAGE

4 A function, `FormOut()`, takes an integer parameter in the range 0 to 999999 and returns a formatted string depending on two other parameter values.

Formatting may incorporate the use of:

- A prefix string to be added before the integer value (e.g. `'$'` or `"Total: "`)
- A comma as a thousand-separator (e.g. `"1,000"`)

The function will be called as follows:

```
        MyString ← FormOut(Number, Prefix, AddComma)
```

Parameter	**Data type**	**Description**
`Number`	`INTEGER`	The positive integer value to be formatted.
`Prefix`	`STRING`	A string that will appear in front of the numeric value. Set to an empty string if no prefix is required.
`AddComma`	`BOOLEAN`	`TRUE` if a comma is required in the formatted string. `FALSE` if a comma is not required in the formatted string.

(a) Fill in the tables to show **two** tests that could be carried out to test **different** aspects of the function.

Give the expected result for each test.

TEST 1

Parameter	**Value**
`Number`	
`Prefix`	
`AddComma`	

Expected return string:

....

TEST 2

Parameter	**Value**
`Number`	
`Prefix`	
`AddComma`	

Expected return string:

....

[4]

(b) Write **pseudocode** for the function `FormOut()`.

Refer to the **Appendix** on pages 18–19 for the list of built-in functions and operators.

[8]

5 A message may contain several hashtags.

A hashtag is a string consisting of a hash character '`#`', followed by one or more alphanumeric characters.

A hashtag may be terminated by a space character, the start of the next hashtag, any other non-alphanumeric character, or by the end of the message.

For example, the following message contains three hashtags:

```
"#Error27 is the result of #PoorPlanning by the #Designer"
```

The hashtags in the message are `"#Error27"`, `"#PoorPlanning"` and `"#Designer"`.

A program is being developed to process a message and extract each hashtag.

A global 1D array of strings, `TagString`, will store each hashtag in a single element.
Unused array elements will contain an empty string. The array will contain 10 000 elements.

A developer has started to define the modules as follows:

Module	**Description**
`GetStart()`	• Called with two parameters: ○ a message string ○ an integer giving the number of the required hashtag. For example, `GetStart(Message, 3)` would search for the third hashtag in the string `Message` • Returns an integer value representing the start position of the hashtag in the message string, or value −1 if that hashtag does not exist
`GetTag()`	• Called with two parameters: ○ a message string ○ an integer giving the hashtag start position within the message • Returns the hashtag or an empty string if the character in the message at the hashtag start position is not `'#'`
`GetIndex()`	• Called with a hashtag as a parameter • Returns the index position of the hashtag in array `TagString` • Returns the value −1 if the hashtag is not present in the array

(a) Write **pseudocode** for the module `GetIndex()`.

.... [6]

(b) Write **pseudocode** for the module `GetStart()`.

The module description is repeated here for reference.

Module	**Description**
`GetStart()`	• Called with two parameters: ◦ a message string ◦ an integer giving the number of the required hashtag. For example, `GetStart(Message, 3)` would search for the third hashtag in the string `Message` • Returns an integer value representing the start position of the hashtag in the message string, or value −1 if that hashtag does not exist

[6]

BLANK PAGE

(c) Write **program code** for the module `GetTag()`.

The module description is repeated here for reference.

Module	**Description**
`GetTag()`	• Called with two parameters: ○ a message string ○ an integer giving the hashtag start position within the message • Returns the hashtag or an empty string if the character in the message at the hashtag start position is not `'#'`

Visual Basic and Pascal: You should include the declaration statements for variables.
Python: You should show a comment statement for each variable used with its data type.

Programming language ....

Program code

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

[8]

Appendix

Built-in functions (pseudocode)

Each function returns an error if the function call is not properly formed.

`LENGTH(ThisString : STRING) RETURNS INTEGER`
returns the integer value representing the length of string `ThisString`

Example: `LENGTH("Happy Days")` returns `10`

`LEFT(ThisString : STRING, x : INTEGER) RETURNS STRING`
returns leftmost `x` characters from `ThisString`

Example: `LEFT("ABCDEFGH", 3)` returns string `"ABC"`

`RIGHT(ThisString: STRING, x : INTEGER) RETURNS STRING`
returns rightmost `x` characters from `ThisString`

Example: `RIGHT("ABCDEFGH", 3)` returns string `"FGH"`

`INT(x : REAL) RETURNS INTEGER`
returns the integer part of `x`

Example: `INT(27.5415)` returns `27`

`MOD(ThisNum : INTEGER, ThisDiv : INTEGER) RETURNS INTEGER`
returns the integer value representing the remainder when `ThisNum` is divided by `ThisDiv`

Example: `MOD(10,3)` returns `1`

`MID(ThisString : STRING, x : INTEGER, y : INTEGER) RETURNS STRING`
returns a string of length `y` starting at position `x` from `ThisString`

Example: `MID("ABCDEFGH", 2, 3)` returns string `"BCD"`

`LCASE(ThisChar : CHAR) RETURNS CHAR`
returns the character value representing the lower case equivalent of `ThisChar`
If `ThisChar` is not an upper-case alphabetic character, it is returned unchanged.

Example: `LCASE('W')` returns `'w'`

`DIV(ThisNum : INTEGER, ThisDiv : INTEGER) RETURNS INTEGER`
returns the integer value representing the whole number part of the result when `ThisNum` is divided by `ThisDiv`

Example: `DIV(10,3)` returns `3`

`NUM_TO_STRING(x : REAL) RETURNS STRING`
returns a string representation of a numeric value.
Note: This function will also work if `x` is of type `INTEGER`

Example: `NUM_TO_STRING(87.5)` returns `"87.5"`

Operators (pseudocode)

Operator	Description
`&`	Concatenates (joins) two strings Example: `"Summer" & " " & "Pudding"` produces `"Summer Pudding"`
`AND`	Performs a logical `AND` on two Boolean values Example: `TRUE AND FALSE` produces `FALSE`
`OR`	Performs a logical `OR` on two Boolean values Example: `TRUE OR FALSE` produces `TRUE`

BLANK PAGE